Supporting Information: Automatic differentiation in quantum chemistry with applications to fully variational Hartree-Fock

Teresa Tamayo-Mendoza,[†] Christoph Kreisbeck,[*,†] Roland Lindh,[‡] and Alán Aspuru-Guzik[*,†]

†*Department of Chemistry and Chemical Biology, Harvard University, 12 Oxford St, Cambridge, MA 02138, USA*

‡*Department of Chemistry-Ångström, The Theoretical Chemistry Programme,*

¶*Uppsala Center for Computational Chemistry, UC3, Uppsala University, Box 518, 751 20, Uppsala, Sweden*

§*Canadian Institute for Advanced Research, Toronto, Ontario M5G 1Z8, Canada*

E-mail: christophkreisbeck@gmail.com; aspuru@chemistry.harvard.edu

A Derivatives of eigenvectors

The differentiation of the matrix diagonalization $\mathbf{D^T A D} = \mathbf{\Lambda}$ is implemented in several libraries as an elementary operation. In the following derivations, we provide a broad outline of the methods and their limitations in differentiating matrix diagonalization for forward and backward modes. For further detail, we refer to Refs.[1–4]

In the case of backward mode, the adjoint of eigenvectors $\overline{\mathbf{A}}$ can be obtained by implicitly

differentiating the eigenvalue problem, resulting in the following expression

$$\overline{\mathbf{A}} \;=\; \left(\mathbf{D}^{-1}\right)^T \left(\overline{\mathbf{\Lambda}} + \mathbf{F} \circ \mathbf{D}^T\mathbf{D}\right) \mathbf{D}^T, \tag{1}$$

where F is zero along the diagonal and $F_{ij} = (\lambda_j - \lambda_i)^{-1}$ for $i \neq j$. Note that F_{ij} diverges for repeated eigenvalues. Therefore, we cannot use backward differentiation for systems with degenerate molecular orbitals.

In the case of forward mode, one method to compute the derivatives $\dot{\mathbf{D}}$ is by finding the appropriate matrix $\mathbf{C}$, such that,

$$\dot{\mathbf{D}} = \mathbf{D}\mathbf{C}. \tag{2}$$

For non-degenerate eigenvalues, $\mathbf{C}$ can be obtained by

$$c_{ij} \;=\; \frac{\mathbf{D}_i^T \dot{A} \mathbf{D}_j}{2(\lambda_i - \lambda_j)} \quad i \neq j \text{ and } c_{ii} = -\frac{1.0}{D_{ii}} \sum_m^n D_{im} c_{mi} \qquad . \tag{3}$$

It is possible to extend this approach to the degenerate case though this requires calculations of higher order derivatives

$$c_{ij} = \frac{\mathbf{D}_i^T A'' \mathbf{D}_j}{2(\dot{\lambda_{ii}} - \dot{\lambda_{jj}})}. \tag{4}$$

In cases in which derivatives of eigenvectors are repeated, there is a similar expression that again requires computing derivatives of the next order. This procedure needs to be repeated up to the point in which all diagonal terms on the nth order derivatives of eigenvalues are distinct. As a result, such implementation will require computations of higher-order derivatives as well as a case-by-case analysis depending on the problem at hand.

An alternative algorithm was proposed by Walter et al.[2] which has been implemented in the the Python library *Algopy*.[5] This library utilizes the univariate Taylor polynomial arithmetic[6,7] to compute higher-order derivatives and applies a general algorithm for the eigenvalue problem. This formalism offers a scheme to compute higher order derivatives of

explicit and implicit functions. For each dependent and independent variable, there is a Taylor polynomial whose coefficients correspond to the value of the variable and its derivatives. With these, we can construct a system of equations that contains different differentiation order to compute the coefficients of the polynomial corresponding to the dependent variable. The set of equations is solved by using the the Newton-Hensel lifting method, also called Newton's method.[6] In our specific case, to cover the degenerate case, these set of equations are computed systematically by blocks of D and Λ of the same eigenvalue or derivative.

B Univariate Taylor Arithmetic

Forward differentiation can be formulated with the Taylor ring arithmetic.[6] This arithmetic allows us to implicitly differentiate a function and compute higher order derivatives. In the following, consider a function $y(t) = F(x(t))$ where $F : \mathbb{R}^n \longmapsto \mathbb{R}^m$, for a given smooth curve

$$x(t) = \sum_{d=0}^{D-1} x_d t^d \tag{5}$$

with $t \in (-\epsilon, \epsilon)$. Using Taylor's theorem we obtain

$$y(t) = \sum_{d=0}^{D-1} y_d t^d + \mathcal{O}(t^D) \tag{6}$$

where

$$y_d := \frac{1}{d} \frac{\partial^d}{\partial t^d} \, x(t)|_{t=0} \tag{7}$$

In this form the coefficients $\mathbb{R}^{D\times n}$ included in $x(t)$ are mapped to the coefficients $\mathbb{R}^{D\times m}$ yielding a approximate expression for $y(t)$. Note that these sets of polynomials defined by

$$\mathcal{P}_d = \left\{ x(t) = \sum_{j=0}^{d-1} x_j t^j \,\middle|\, x_j \in \mathbb{R} \right\} \tag{8}$$

form a commutative ring for every order $d > 0$. This allows us to obtain Taylor polynomials from the binary operations of addition, multiplications and subtraction operations. Furthermore, we can apply the rules of multivariate calculus to any continuous and d-time differentiable function F. These functions and their derivatives can be mapped to their corresponding Taylor coefficients with the extension function $E_D(F)$, $E_D(F) : \mathcal{P}_D^n \longmapsto \mathcal{P}_D^m$, defined by,

$$\begin{aligned} [y]_D &= E_D(F)([x])_D = \sum_{d=0}^{D-1} y_d t^d \\ &= \sum_{d=0}^{D-1} \frac{1}{d!} \frac{\partial^d}{\partial d^t{}_d} F\left(\sum_{d=0}^{D-1} x_d t^d\right)\Bigg|_{t=0} T^d \end{aligned} \tag{9}$$

with $[x]_D \equiv [x_0, x_1, ..., x_{D-1}]$ and $[x]_{d:D} \equiv [x_d, x_1, ..., x_{D-1}]$. Furthermore, this extended function follows the rules of composite functions, $F(x) = (H \circ G)(x)$ such that,

$$E_D(F) = E_D(H) \circ E_D(G).$$

This relation sets the foundation for forward differentiation that allows us to use the chain rule with a set of elementary operations. Furthermore, we can extend this formalism to implicit equations $0 = F(x, y) \in \mathbb{R}^M$. For a given Taylor polynomial $[x]_D$, we can find the polynomial $[y]_D$ which is defined by a system of equations, up to order D

$$0 \overset{\text{D}}{=\!=} E_D(F)([x]_D, [y]_D), \tag{10}$$

where $[x] \overset{\text{D}}{=\!=} [y]$ if $x_d = y_d$ for $d = 0, ..., D-1$. Once we know the coefficients $[y]_D$, we can solve the next E orders where $1 \leq E \leq D$, using the Newton-Hensel lifting method.[6] The method solves the next set of equations based on the previous orders, such that

$$0 \overset{\text{D+E}}{=\!=} E_{D+E}(F)([x]_{D+E}, [y]_{D+E}). \tag{11}$$

If the function F is differentiable and $F_y(x_0, y_0)$ is invertible, then the new coefficients are calculated by $[\Delta y]_E = [\Delta y]_{D+E} - [\Delta y]_E$, which can be computed by the expression

$$[\Delta y]_E = [F_y]_E^{-1}[\Delta F]_E. \tag{12}$$

where $E_{D+E}(F)([x]_{D+E}, [y]_{D+E}) \overset{\text{D+E}}{=\!=} [\Delta F]_E^{-1} T^D$ and $[F_y]_E = E_E\left(\frac{\partial F}{\partial y}\right)([x]_E, [y]_E)$. These equations allow us to employ the Newton-Hensel method to compute the Taylor coefficients of f[y]. This method solves iteratively the system of equations eq. (11), by either doubling the number of coefficients following Newton's method, or solving the coefficient once at a time, using sequentially the lifting Newton-Hensel algorithm.

References

(1) Giles, M. B. In *Advances in Automatic Differentiation*; Bischof, C. H., Bücker, H. M., Hovland, P., Naumann, U., Utke, J., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2008; pp 35–44.

(2) Walter, S. F.; Lehmann, L.; Lamour, R. On evaluating higher-order derivatives of the QR decomposition of tall matrices with full column rank in forward and reverse mode algorithmic differentiation. *Optim. Method. Softw.* **2012**, *27*, 391–403.

(3) Walter, S. Structured higher-order algorithmic differentiation in the forward and reverse mode with application in optimum experimental design. Ph.D. thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, 2012.

(4) Nelson, R. B. Simplified calculation of eigenvector derivatives. *AIAA Journal* **1976**, *14*, 1201–1205.

(5) Walther, A.; Griewank, A. In *Combinatorial Scientific Computing*; Naumann, U.,

Schenk, O., Eds.; Chapman-Hall CRC Computational Science, 2012; Chapter 7, pp 181–202.

(6) Griewank, A.; Walther, A. *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation, second edition*; SIAM, Philadelphia, 2008.

(7) Bücker, H. M., Corliss, G. F., Hovland, P. D., Naumann, U., Norri, B., Eds. *Automatic Differentiation: Applications, Theory, and Implementations*; Lecture Notes in Computational Science and Engineering; Springer: New York, NY, 2005; Vol. 50.